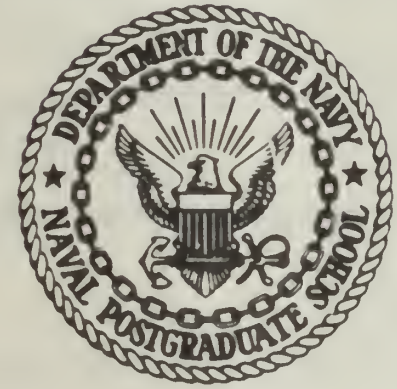
NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

SOFTWARE COST ESTIMATION
THROUGH
BAYESIAN INFERENCE OF SOFTWARE SIZE

by

Park, In Kyoung

September 1985

Thesis Advisor: Dan C. Boger
Co-Advisor: Michael G. Sovereign

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A161 184

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
Software Cost Estimation Through Bayesian Inference of Software Size

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis
September, 1985

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Park, In Kyoung

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93943-5100

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93943-5100

12. REPORT DATE
September, 1985

13. NUMBER OF PAGES
70

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Software Cost Estimation; Bayesian Inference; SLIM; Software Life-Cycle; Rayleigh Distribution.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
It is important for the program manager to understand the software life-cycle and development process. Multiple models of the life-cycle are examined and compared with the DoD software life-cycle. The Rayleigh equation and the resulting difficulty gradient equation of the SLIM software model are very powerful techniques for estimating development time, effort, and cost. To estimate the size of the new project a Bayesian inference —

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. technique is proposed. This technique is then applied using the SLIM model and the QSM data base.

Approved for public release; distribution is unlimited.

Software Cost Estimation through Bayesian Inference of Software Size

by

Park, In Kyoung
Lieutenant, Republic of Korea Navy
B.S., Korea Naval Academy, 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

ABSTRACT

It is important for the program manager to understand the software life-cycle and development process. Multiple models of the life-cycle are examined and compared with the DOD software life-cycle. The Rayleigh equation and the resulting difficulty gradient equation of the SLIM software model are very powerful techniques for estimating development time, effort, and cost. To estimate the size of the new project a Bayesian inference technique is proposed. This technique is then applied using the SLIM model and the QSM data base.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I wish to thank Professors Dan C. Boger of the Administrative Science Department and Michael G. Sovereign of the Operations Research Department for their academic guidance as well as numerous practical comments.

I would also like to thank the staff of Quantitative Software Management Inc.

I also want to express my great thanks to Lawrence H. Putnam, President of QSM Inc., for his assistance in obtaining this data base and in teaching me the SLIM model.

I wish to express my appreciation to all those who graciously provided assistance, information, and advice in preparation of this thesis.

I. INTRODUCTION

This thesis is an attempt to understand the software cost estimating process as viewed by a program manager during the early phases of system development.

Software cost estimators have problems because of their inability to estimate the size of new programs. Historically-based methods are difficult to use because the estimator is unable to assess skills, tools, complexity, and environment of past developments and of the new software. The results are severe cost overruns, schedule slippage, and lack of credibility. [Ref. 1]

In software estimating models, software size is the key parameter influencing cost. All the proven estimating techniques begin with an estimate of the size of the software package and then at various levels of sophistication produce an estimate of cost or time based on size and calculated or derived productivity factors.

A quick, reliable estimate of size could provide a better cost estimate for planning purposes. The program manager's concern is time, cost, and performance of the total system. He needs to present this information to higher DOD management levels during the early planning phase.

This paper is an investigation of ways in which an estimator may be able to estimate the size of a new project, based on an existing data base of close to 1,000 systems. Breakouts have been made of that data base by application type and the average size and the standard deviation of each of these applications is calculated. In the early phase of system development, a reasonable goal would be to estimate time within 15 percent and effort within 30 percent of actual.

The program manager should have a detailed understanding of the process that developed the cost estimate and should have some traceability to the variance or sensitivity of the estimating component. The use of automated software models for estimating has become popular in industry and DOD. Program managers can compare different models for alternative estimates.

All the variables of the software equation are subject to some degree of uncertainty and the program manager must have a means of taking this into account in an effort to development risk profiles. Putnam [Ref. 2] suggests that risk analysis is probably the most important aspect of any software system analysis. In an uncertain process, risk analysis measures that uncertainty. This leads to strategies tc minimize risk. The program manager should perform risk analysis and determine the probability of meeting schedule/cost.

This paper uses SLIM (Software LIfe-cycle Management). The SLIM model uses some of the most powerful tools of analysis available today -- linear programming, Monte Carlo simulation, and algorithms from the PERT methodology -- to produce the best possible solution to the software estimating problem. [Ref. 1]

The program manager predicts and controls schedule and costs. It is important for the program manager to recognize potential problems early and take effective corrective action.

II. SOFTWARE LIFE-CYCLE AND DEVELOPMENT PROCESS

The software life-cycle may be said to start in the perception of a need and to terminate when the system is retired as obsolete. The program manager has the responsibility from start to end for the project. The program manager must understand the software life-cycle and its development process. A lack of understanding of the process of software development (activities, phases, and milestones) causes a poor estimate. The software development process can be subdivided into sequential and overlapping phases.

Figure 2. 1 [Ref. 2] depicts the total system milestones in relationship with four software cost models considered. The DOD subdivides the life cycle of an automated information system into five broad phases: [Ref. 3] Mission Analysis Project Initiation, Concept Development, Definition Design, System Development, Deployment Operation.

Putnam [Ref. 4] divides it as: Systems definition, Functional design specification, Development(design and coding, test and validation), Operation and maintenance. The development process is divided as follows.

PDR - Preliminary Design Review. This is the earliest time that a formal review of the functional design specifications can be expected to be satisfactory enough to continue into the next phase of development. Functional design and (high level) system engineering is essentially complete.

CDR - Critical Design Review. This is a review of the detailed logic design for each element of system. The design consists of flow charts, HIPO[1] diagrams, Pseudo-code

---

[1]HIPO (Hierarchy-Process-Input-Output) was developed at IBM as design representation schemes for top-down software development and contained a visual table of contents, a set of overview overview diagrams, and a set of detail diagrams.

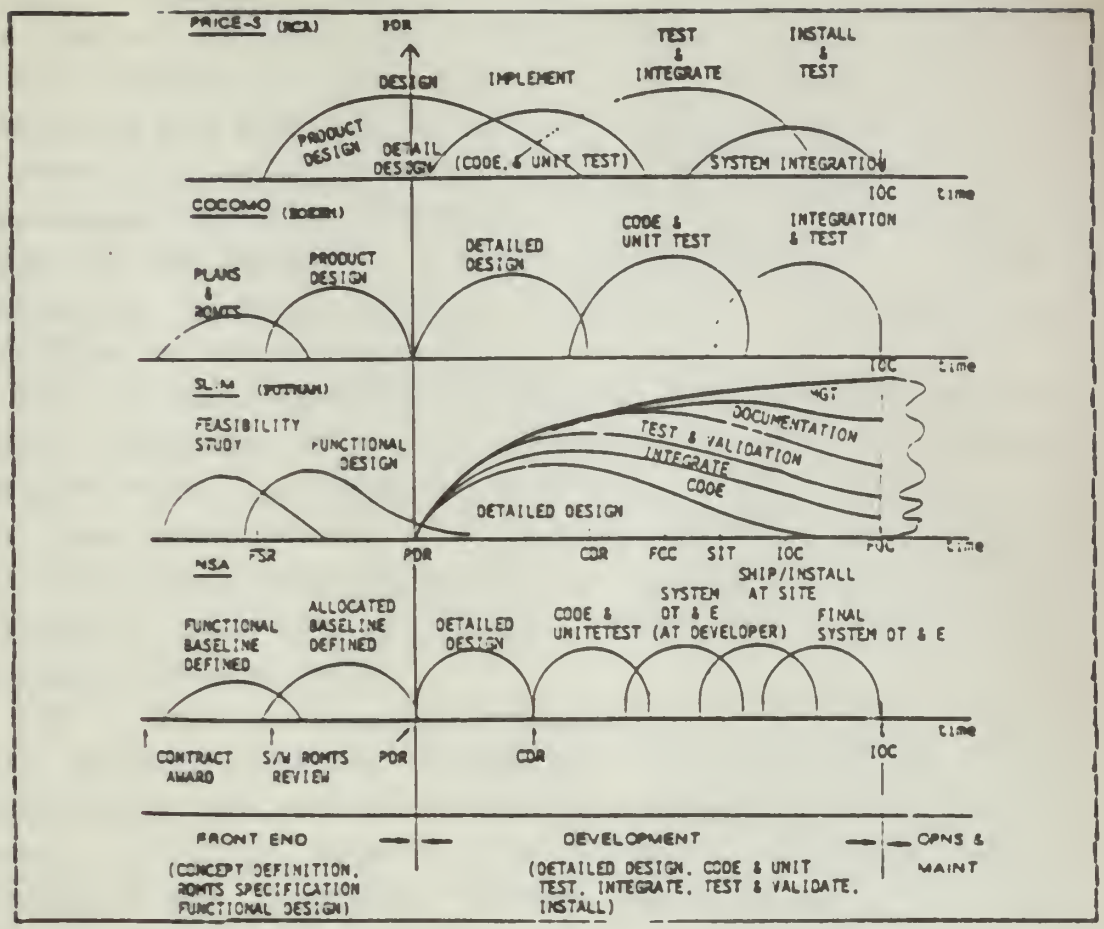


Figure 2.1 Software Life-Cycle.

logic, or equivalent. It is held when design and coding are separated by management decision as by MIL STD (military standards). Coding cannot start until after a successful CDR under this philosophy. There must be sufficient design to start coding.

FCC - First Code Complete. In a top-down, structured design and coding environment, FCC is the time at which all the units of code have been written, the units have been peer and management checked, successfully compiled and run as units, and thought to be satisfactory end product code. It is then entered into a library of completed code. (Note: coding will continue thereafter as rework of these modules.)

SIT - Systems Integration Test. This is the earliest time that all elements and subsystems have been put together and the system can work together as a complete integrated package and can demonstrate that in a formal system test.

UOST- User Oriented System Test. Following correction of deficiencies resulting from SIT, UOST is the first time that a test of the system in a full user environment -- target machine and operating system, real data, real operating conditions -- can be conducted.

IOC - Initial Operational Capability or start of installation, depending on the environment. This is a careful, tentative first use under rigid control. Often this is a first site installation in a live environment with anticipated later multi-site deployment, or the start of operation in parallel with the predecessor system in a single site replacement environment.

FOC - Full Operational Capability. Here the system meets specified quality standards sufficiently well that organizations will use it in everyday routine mission operations. (In SIIM, that is a 95 percent reliability level; calibration and technology factors are normalized to this reliability level.)

Boehm [Ref. 5] divides the cycle into feasibility, plans and requirements, product design, programming, integration and test, and maintenance. The primary emphasis of his COCOMO[2] model is on the development portion of the life-cycle which COCOMO defines as starting " at the beginning of the product design phase and ending at the end of the software integration and test phase ". [Ref. 5]

---

[2]COnstructive COst MOdel, a hierarchy of three increasingly detailed models (basic, intermediate, detail) which range from a single macro-estimation scaling model as a function of product size to a micro-estimation model with a three-level work breakdown structure and a set of phase-sensitive multipliers for each cost driver attribute.

The PRICE-S software model is one of the family of RCA cost predicting models. PRICE-S divides the software development cycle into three phases: design, implementation, and test and integration.

The question facing the program manager is what to invest in and what is its payoff. Boehm [Ref. 5] developed a value-of-information guideline with the following five conditions:

1. There exist attractive alternatives whose payoff varies greatly, depending on some critical state of nature.
2. The critical states of nature have an appreciable probability of occuring.
3. The investigations have a high probability of accurately identifying the occurrence of critical states of nature.
4. The required cost and schedule of the investigations do not overly curtail their net value.
5. There exist significant side benefits derived from performing the investigations.

The major difficulty with these guidelines is determining what are the critical states of nature. Boehm implies some of these states in his definition of feasibility phase and plans and requirements phase [Ref. 5]:

Feasibility phase: How much should we invest in information system (user questionnaires and interview , current-system analysis, workload characterizations, simulations, scenarics, prototypes) in order that we converge on an appropriate definition and concept of operation for the system we plan tc implement?

Plans and requirements phase: How rigorously should we specify requirements? How much should we invest in requirements validation activities(automated completeness, consistency, and traceability checks, analytic models, simulations, prototypes) before proceeding to design and

develop a software system? It is important for the program manager to know the early phases of system development.

Boehm [Ref. 6] points out the uncertainty of cost estimation during the early project phase. Fig 2.2 shows the accuracy within which software cost estimation can be made, as a function of the software life-cycle phase ,or of the level of knowledge we have of what the software is intended to do. At the beginning of the feasibility phase, the relative range of software cost estimates is roughly a factor of four[3] on either the high or low side which is not surprising based on the limited knowledge available at this point. The estimation uncertainty is reduced to from .5 to 2 after the concept of operation has been defined. After completion of a requirement specification, the estimation range is .67 to 1.5. Gradually, the estimation range becomes smaller until we finally arrive at acceptance.

Fairley's [Ref. 7] suggested life-cycle approach divides into four models: The phased model, cost model, prototype life-cycle model, and successive versions model. Costs incurred within each phase include the cost of performing the process and preparing the products for that phase, plus the cost of verifying that the products of the present phase are complete and consistent with respect to all previous phases. Fairley also suggests there are some reasons for developing a prototype: (1) to illustrate input data formats, message, reports , and interactive dialogues for the customer; (2) to explore technical issues in the proposed product; and (3) in situations where the phased model or analysis -> design -> implementation is not appropriate. Product development by the method of successive versions is an extension of prototyping in which an initial product skeleton is refined into increasing levels of

---

[3]The ranges are intended to represent 80 percent confidence limits, that is , " within a factor of four on either side, 80 percent of the time."

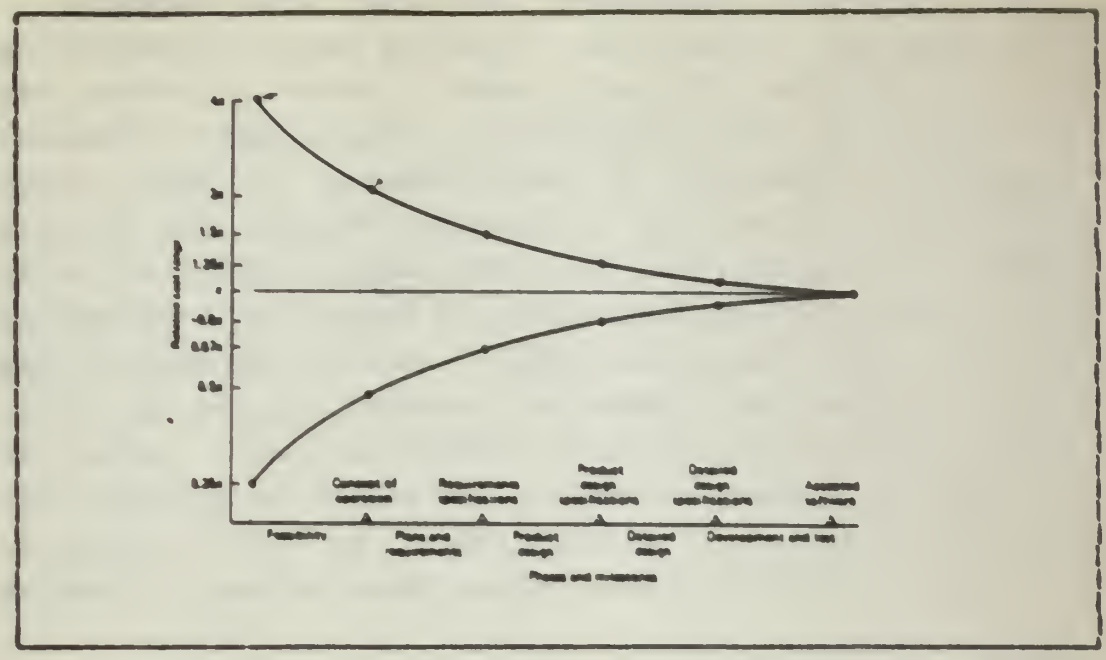


Figure 2.2 Software Cost Estimation Accuracy vs. Phase.

capability.

Putnam [Ref. 8] suggests that a good life cycle model should possess these characteristics:

1. Consider all activities and phases.
2. Relate management parameters to management responsibilities.
3. Be adaptive to actual project data and requirements changes(i.e. must be time-varying or dynamic)
4. Provide engineering accuracy.
5. Provide sensitivity profiles.
6. Be phenomenologically based.
7. Relate produce to resource consumption (both statically and dynamically) the technology being applied.
8. Be capable of future growth.
9. Be able to adequately treat known and future system types and development environments.

Software life cycle and development processes offer a structured means for planning, developing, and controlling the software project. Boehm [Ref. 9] suggests that the tradeoffs between lower development costs and lower life cycle costs are characterized by the modern programming practice and required reliability cost estimating relationships for software development and maintenance. The program manager must understand the software life-cycle and development process of multiple models and compare with DOD life-cycle. The program manager must predict and control schedule and costs. Also the program manager must recognize potential problems early and take effective corrective action. It is important for the program manager to minimize the cost/schedule impact of requirements changes.

III. ESTIMATING TECHNIQUES AND COST ATTRIBUTE FACTORS

In order for the program manager and his support team to evaluate a software cost proposal, they must have a detailed understanding of the process that developed the cost estimate and should have some traceability to the variances or sensitivity of the estimating components.

The object of software cost estimation is to determine what resources (manpower, computer time, and elapsed time) will be needed to produce the software associated with the project. Stanley [Ref. 10] listed some reasons for not obtaining a good estimate as:

1. A lack of understanding of the process of software development.
2. A lack of understanding of the effects of various technical and management constraints.
3. A view of that each project is unique, which inhibits project to project comparisons.
4. A lack of historical data against which the model can be checked and for calibration.

Boehm [Ref. 5] suggests that a good software model should possess the following characteristics:

1. Definition: Can you tell what it is estimating? Will different people give similar factor rating?
2. Fidelity: Are the actuals close to the estimates?
3. Detail: Does it give (accurate) phase and activity breakdowns?
4. Constructiveness: Can you tell why it gives the estimates it does? Does it help you understand the software job?
5. Objectivity: Is it hard to jigger the model to get any result you want?
6. Stability: Do small input changes produce small output

changes?

7. Scope: Does the model cover your class of software project?
8. Easy to use: Are the model inputs, and outputs easy to understand and specify?
9. Prospectiveness: Does it depend on information not known until later?
10. Parsimony: Does it use redundant or unnecessary factors?
11. Availability: Can I get access to the model?

The program manager compares various cost models for alternative estimates. The model should allow for the use of historic data in calibration for a particular organization and type of software. A good software cost estimation model should cover software engineering issues which arise throughout the software life cycle.

A. TECHNIQUES

According to Stanley [Ref. 10], most cost models use one or both of the following equations. The first is called the cost equation:

$$E = a \times S^{b}$$

where E = effort needed, s = size of program is terms of some measure of lines of code and a and b are chosen by curve fitting on a historical database. Different values of a and b are appropriate to different organizations, project types, units of measurement of E and S, and items included in the the estimates.

The second equation is called the general summing equation:

$$Z = \sum_{i=1}^{n} a_i f_i = a_1 f_1 + a_2 f_2 + \ldots\ldots + a_n f_n$$

where the a_i are input parameters derived from the description of software characteristics and the characteristics of the development environment, and the value of f_i are chosen by curve fitting on a suitable historic database. Table 1 [Ref. 7] shows a comparison of effort and development time equations according to software size. Some models provide equations to estimate total man months of effort, MM, in terms of the number of thousands of delivered source instructions, KDSI. Also the development time for software project, TDEV, measured in terms of MM. A software project's cost can be obtained by multiplying the effort in man months by the cost per man month.

TABLE 1

COMPARISON OF EFFORT AND TIME EQUATIONS [Ref. 7]

Effort equation MM =	Development time TDEV =	Author
5.2 (KDSI) **0.91	2.47 (MM) **0.35	Walston
4.9 (KDSI) **0.98	3.04 (MM) **0.36	Nelson
1.5 (KDSI) **1.02	4.38 (MM) **0.25	Freburger
2.4 (KDSI) **1.05	2.50 (MM) **0.38	Boehm
3.0 (KDSI) **1.12	2.50 (MM) **0.35	Boehm
3.6 (KDSI) **1.20	2.50 (MM) **0.32	Boehm
1.0 (KDSI) **1.40	-	Jones
0.7 (KDSI) **1.50	-	Halstead
28 (KDSI) **1.83	-	Schnieder

Thibodeau [Ref. 11] states that parametric models may be divided into three classes:

1. Regression model: The parameters to be estimated are mathematically related to a set of input parameters. The parameters of the hypothesized relationships are arrived at by statistical analysis and curve fitting on an appropriate historical database.
2. Heuristic models: Here observation and interpretation of historic data are combined with supposition and

experience.

3. Phenomenological model: This is based on a hypothesis that the software development process can be explained in terms of some more widely applicable process or idea.

The first model class seems to be the approach used in most cost modeling. This class includes the Aerospace, Doty, Farr, Zagorski, and Telecote models. The second model class seems to be the type used in preparing a proposal where detailed WBS elements are prepared and summed for the total cost. The Boeing and PRICE S models along with the DOD MICRO Procedure and the Wolverton model have been termed heuristic. An example of the last class is the Putnam model. [Ref. 11]

Stanley [Ref. 10] suggests a general pattern followed by all the models:

1. Estimate software size
2. Convert size estimate to labor estimate
3. Adjust estimate for special project characteristics
4. Divide the total estimate into the different project phases
5. Estimate non-technical labor costs
6. Sum the cost

Most models start from an estimate of project size. Some models convert from size to labor, others go directly from size to money estimates. The effective estimate is an adjustment of the basic estimate intended to take account of any special project characteristics which make it dissimilar to the pattern absorbed in the underlying historic database. Each model which deals with a project's schedule makes assumptions about the allocation of effort in different project phases.

B. SOME SPECIFIC TECHNIQUES

From the QSM[4] data base, we perform curve fitting to get the relationship of software size versus development time and effort using the regression method. Despite the wide ranges, the development time and effort correlate very well with the number of source statements. Figure 3.1 shows size versus development time from the curve-fitting method.

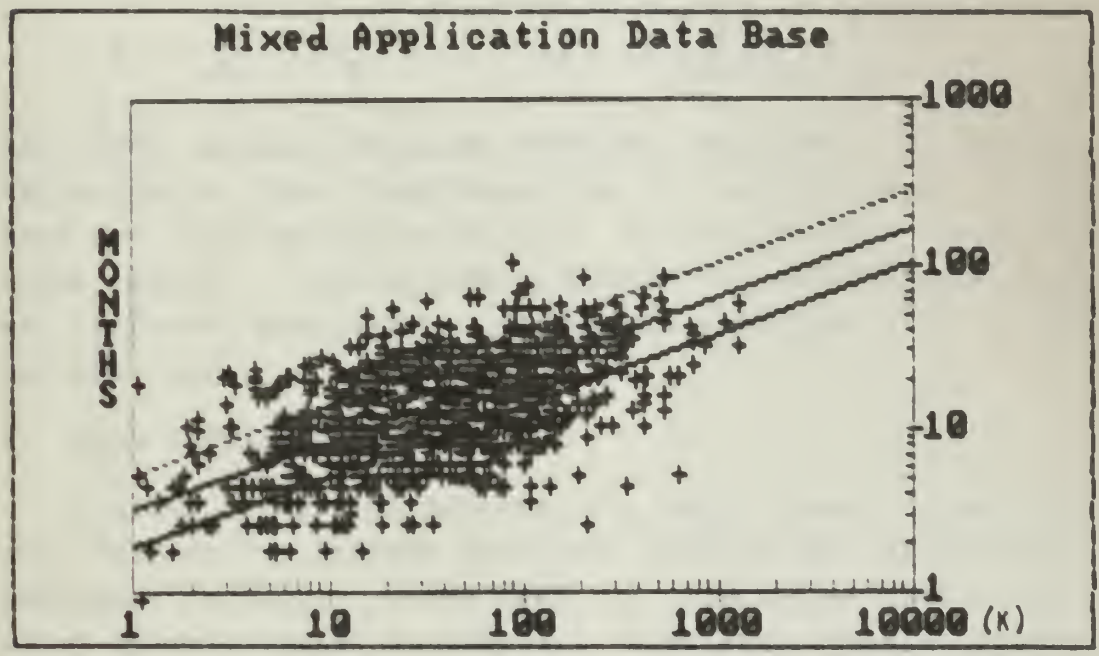


Figure 3.1 QSM Database Software Size vs. Development Time.

To estimate software size, the PERT[5] method is suggested. The PERT equations estimate the expected size (ES) and standard deviation (σ) of each subsystem as

$$ES_i = (a_i + 4m_i + b_i) / 6 \quad , \quad \sigma_i = (b_i - a_i) / 6$$

where

---

[4]Quantitative Software Management, Inc.

[5]Program Evaluation and Review Technique.

a_i = the lowest possible size of the software subsystem
m_i = the most likely size of the subsystem
b_i = the highest possible size of the subsystem

The estimated total software size (S) and standard deviation (σS) are then

$$S = \sum_{i=1}^{n} ES_i \quad , \quad \sigma S = \left(\sum_{i=1}^{n} \sigma_i^2 \right)^{1/2}$$

This PERT sizing technique requires more work to break up the software into subsystems and to estimate most likely size for each subsystem as well as to eliminate upper and lower limits. PERT estimates should be made by knowledgeable software designers, preferably those who will do the specific work.

C. COST ATTRIBUTE FACTORS

Stanley [Ref. 10] defines two major area of software cost drivers: project specific factors and organization dependent factors. Boehm [Ref. 9] identifies five factors which closely match Stanley's. Size attributes, program attributes and computer attributes fall into Stanley's project specific factor. Personnel attributes and project attributes fall into Stanley's organization-dependent factors. Wolverton [Ref. 12] suggests that top-level characteristics are parameters which can be classified into software structural parameters and project financial parameters. Sortware structural parameters may be divided size, program attributes, hardware attributes, project attributes, environmental attributes. Project financial parameters are divided into direct labor charges, overhead, other direct charge, general and administrative expense, and fee.

Bruce and Pederson [Ref. 13] divide cost drivers into four categories: requirement factors, product factors, process factors, and resource factors.

1. Requirement Factor

Two factors that can have a major impact on the ultimate cost of a software product are (1) the quality of the specifications and (2) the stability of the requirement.

a. Quality of Specifications

A good definition of requirements is the cornerstone of a well-defined, well-understood, and well-costed software development. Incomplete requirement definition is a major cause of cost overruns.

b. Stability of Requirements

The responsibility of the project manager is to understand the software requirements and to point out that changes in the requirement baseline are changes. The project manager should then define the cost and/or schedule impact so that the change can be given a fair evaluation.

2. Product Factors

a. Software Size

A favorite measure for software system size is lines of operational code or deliverable code. Parametric cost estimating models relate cost in some way to the size estimate.

b. Difficulty

After considering the relative difficulty of various software systems and functions, the cost estimator must consider the difficulty of the specific application and its heritage to complete the analysis of the difficulty factor.

c. Reliability

Reliability requirements determine the thoroughness of testing needed before software can be used in the intended operational environment. There are four major criteria for determining the reliability of software program. It must (1) provide continuity of operation under nonnominal conditions; (2) utilize uniform design, implementation techniques, and notation; (3) yield the required precision in calculations and outputs; and (4) be implemented in an understandable manner.

d. External Interface

Especially beware of special purpose hardware interfaces and sophisticated user interface.

e. Language Requirement

The use of design languages has been shown to help decrease the time required in programming and to provide well-structured and documented code.

f. Documentation Requirement

Both customer-specified documentation requirements and documentation required by the project management as part of the selected development approach must be considered.

3. Process Factors

The software costing factors associated with the development process, such as management structure, management controls, development methods, tools, use of available software, and data base methods.

a. Management Structure

Management structure incorporates the effects of various company policies with respect to allocating costs for certain non-project management personnel as a direct charge to projects.

b. Management Control

This covers the cost of project support in such areas as management information processing, scheduling support, administrative support, and clerical support.

c. Development Methods

This factor attempts to quantify the impact of various development methods. The development methods of interest include such approaches as top-down design and testing, structured programming, use of chief programmer teams, and use of the structured walk-throughs.

d. Tools

The program manager must consider how the software will be developed, tested, and maintained and what tools will be needed to accomplish these tasks. For systems developed for large-scale computer, a host of complier, data base managers, editors, display interface package, flow chart package, plot package, utility routines, and test data generation tools are generally available. The costs associated with tools are a function of the tool complexity, use, features, and, maturity.

e. Available Software

A significant reduction in project can be achieved through the use of existing software. The costs of modifying the existing software can then be determined subjectively.

f. Data Base

The size, complexity, and special file access requirements for the data base are extremely important parameters in deriving an accurate software development estimate.

4. Resource Factors

Development costs for a given software package may vary substantially, depending on such factors as experience of available people, quality of project staff, and availability of development computer resources.

a. Number of People

The major contributor to the reduction in productivity associated with projects that require larger staffs is the increase in time needed for communication between people.

b. Experience of People

The program manager must consider the general level of experience and skill required for various project assignment and incorporate appropriate labor rates into the estimate.

c. Personnel Performance

Since software development is an analytical, sometimes creative activity requiring abstract reasoning, individual productivity variations are to be expected. Productivity assessment is extremely important because cost estimation generally is reduced to deriving a productivity figure per unit of effort per person for a given skill category.

d. Availability of Computing Resources

For batch development systems there is a linear relationship between turnaround time and testing costs. For interactive development systems, significant development efficiencies are often realized.

e. Suitability of Computing Resources

The asymptotic effect on development costs as the hardware speed and memory size constraints are approached has been demonstrated in batch, real-time, airborne, military, and commercial system.

f. Elapsed Time

The amount of calender time available to develop a software product is intimately related to the ultimate development cost. Below the threshold, the increased staff required to accomplish the job in a shorter period has the opposite of the desired effort.

Table 2 [Ref. 6] shows the various size, program, computer, personnel, and project attributes used by each model to determine software costs.

Numerous factors influence the cost of a software development project and each should be evaluated during cost estimation to ensure that proper weighting is applied to the cost estimates. Current models differ in the factors that are required as specific inputs. Many different factors may be subsumed in a single parameter in some models, particularly the model subjective parameters. The program manager should assess existing personnel/facility resources and determine future requirements. Also the program manager measures overhead and determine the efficiency of resource allocations in multiple development task projects.

TABLE 2

COST DRIVERS USED IN VARIOUS COST MODELS [Ref. 6]

GROUP	FACTOR	SDC 1965	TRW 1972	PUTNAM SLIM	DOTY	RCA PRICE S	IBM	BOEING 1977	GRC 1979	COCOMO	SOFCOST	DSN	JENSEN
SIZE ATTRIBUTES	SOURCE INSTRUCTIONS			x	x		x	x		x	x	x	x
	OBJECT INSTRUCTIONS	x	x		x	x							
	NUMBER OF ROUTINES	x				x					x		
	NUMBER OF DATA ITEMS						x			x	x		
	NUMBER OF OUTPUT FORMATS								x			x	
	DOCUMENTATION				x		x				x		x
	NUMBER OF PERSONNEL			x			x	x			x		x
PROGRAM ATTRIBUTES	TYPE	x	x	x	x	x	x	x			x		
	COMPLEXITY		x	x		x	x			x	x	x	x
	LANGUAGE	x		x				x	x		x	x	
	REUSE			x		x		x	x	x	x	x	x
	REQUIRED RELIABILITY			x		x				x	x		x
	DISPLAY REQUIREMENTS				x						x		x
COMPUTER ATTRIBUTES	TIME CONSTRAINT		x	x	x	x	x	x	x	x	x	x	x
	STORAGE CONSTRAINT			x	x	x	x		x	x	x	x	x
	HARDWARE CONFIGURATION	x				x							
	CONCURRENT HARDWARE DEVELOPMENT	x			x	x	x			x	x	x	x
	INTERFACING EQUIPMENT S/W										x	x	
PERSONNEL ATTRIBUTES	PERSONNEL CAPABILITY			x		x	x			x	x	x	x
	PERSONNEL CONTINUITY						x					x	
	HARDWARE EXPERIENCE	x		x	x	x	x		x	x	x	x	x
	APPLICATIONS EXPERIENCE		x	x		x	x	x	x	x	x	x	x
	LANGUAGE EXPERIENCE			x		x	x		x	x	x	x	x
PROJECT ATTRIBUTES	TOOLS AND TECHNIQUES			x		x	x	x		x	x	x	x
	CUSTOMER INTERFACE	x					x				x	x	
	REQUIREMENTS DEFINITION	x			x		x				x	x	x
	REQUIREMENTS VOLATILITY	x			x	x	x		x	x	x	x	x
	SCHEDULE			x		x				x	x	x	x
	SECURITY						x				x	x	
	COMPUTER ACCESS			x	x		x	x		x	x	x	x
	TRAVEL/MOSTING/MULTI SITE	x			x	x					x	x	x
	SUPPORT SOFTWARE MATURITY									x		x	
CALIBRATION FACTOR				x		x				x			
EFFORT EQUATION	$MM_{NOM} = C(KDSI)^{E} \times \cdot$		1.0		1.047		0.91	1.0		1.05-1.2		1.0	1.2
SCHEDULE EQUATION	$TD = C(MM)^{E} \times \cdot$						0.35			0.32-0.38		0.366	0.333

It is important for the program manager to develop reliable estimates by performing parallel calculations on multiple models.

IV. SLIM

The SLIM model is a comprehensive software cost estimating package produced by Quantitative Software Management Inc. SLIM is an aid for effectively managing software development. Using engineering techniques, SLIM provides cost, time, personnel, and machine estimates for developing computer software systems. SLIM identifies limiting constraints that affect development plans. Confidence levels and risk factors are calculated to provide the manager with the data needed to make decisions on cost, schedule, effort, quality, manloading, and cash flow. The SLIM , software costing and management system does these things [Ref. 2]:

1. Determines the size of the system in source statements
2. Estimates the people, cost and schedule for the project
3. Projects cashflow over the life cycle
4. Identifies limiting constraints on manpower and schedule
5. Obtains risk projections for cost and schedule
6. Updates estimates from the real data once development is underway
7. Dynamically adjusts for requirements changes to give new cost and schedule

The SLIM model is claimed to be valid for any system where at least two of the following four criteria apply:

1. 5000 lines of code or greater
2. $100,000 or more in development costs
3. Peak manpower of 3 people or more
4. Development time of 6 months or longer

Over sixty large organizations such as DOD, IBM, and GTE have used the SLIM package. SLIM's accuracy has been tested for over 1300 systems of all types in the United States and

Europe, International Data Corp., comments, "SLIM is the single most useful management tool for software development that I am aware of at this time" (Ref. 2).

A. SOFTWARE EQUATION

The life-cycle curve can be mathematically represented by the Rayleigh equation, a mathematical form from the Weibull family of reliability functions. Putnam (Ref. 1) has studied the manpower versus time pattern of several hundred medium to large scale software development projects of different classes. These projects all exhibit a similar life cycle pattern of behavior - a rise in manpower, a peaking and a falling off. The Rayleigh equation is:

$$y = (K / t_d^2)\ t\ \exp(-t^2 / 2 t_d^2)$$

K = life-cycle effort in man-years(MY), man-months(MM),

t_d = development time in years(Y), or months(M)

y = manpower in man-years, man-months, or just plain countable at any instant in time

t = the elapsed time in years or months.

Putnam (Ref. 10) points out why the Rayleigh equation is a good software function:

1. It is a management-oriented function
   - area is proportional to cost
   - time parameter(t_d) proportional to schedule
   - curve is manpower at any time and is proportional to spending rate
2. Initial slope is the manpower build-up rate
3. Minimum time for a project is related to maximum manpower acceleration - minimum time becomes predictable
4. Code production rate for system is Rayleigh-like
5. Error rate is Rayleigh-like

6. Rayleigh management parameters can be easily turned into a linear program with important management constraint(manpower, cost, schedule) properly considered to yield constrained optimal solutions. Figure 4.1 [Ref. 2] shows the Rayleigh function as a management tool.

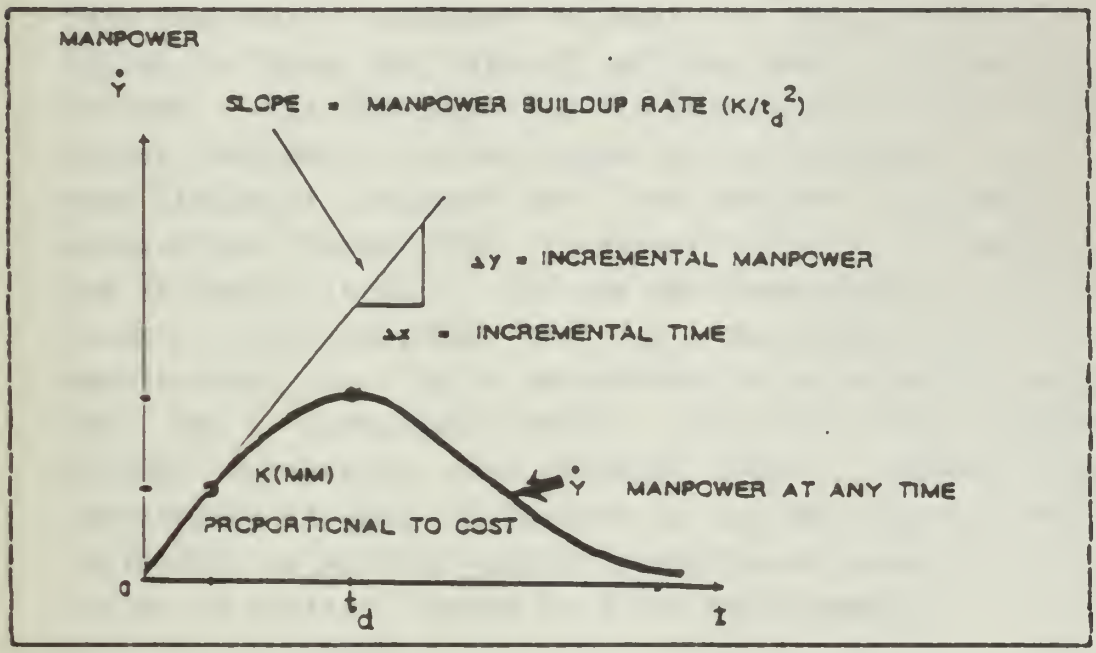


Figure 4.1 Rayleigh Function as Software Management Tool.

The Rayleigh equation is linearized by taking logarithms,

$$Ln(\dot{y}/t) = Ln(K / t_d^2) + (-1 / 2 t_d^2) \quad t^2 \quad .$$

Upon plotting this equation in terms of manpower applied to a system over time squared, a straight line is provided in which $Ln(k / t_d^2)$ is the intercept and $(-1 / 2 t_d^2)$ is the slope. Putnam [Ref. 8] performed this operation for one hundred systems and found that the argument of the intercept (K / t_d^2) has a most interesting property. Putnam

[Ref. 1] suggests that the ratio K / t_d^2 represented the difficulty, D, in terms of the programming effort and the time to produce it. By taking the gradient of D, he gets

$$|\nabla D| = K / t_d^3 = \text{Constant}$$

This difficulty gradient reflects the organizational capability in doing that type of work for which the constant was derived. The difficulty in the development of the software varies inversely with the value of the gradient, i.e, the most difficult projects have the smallest gradient. From observation, Putnam [Ref. 2] suggests $|\nabla D|$ is quantized at the following levels: 7.3 for new developments with interfaces to other programs; 14.7 for a new stand-alone development project; 26.9 for a re-building of an existing program; 55.0 for a composite system1; and 89.0 for a composite system2 containing much existing code. These values's uncertainty is about 15 percent of the base value. When the difficulty is plotted against productivity rate, PR[6], for a number of systems, Putnam gets the relationship

$$PR = C_n D^{-2/3}$$

C_n = productivity constant

The area under the coding rate, $\dot{S}_S$, curve[7] is the total quantity of final end product source statements that will be produced by time t [Ref. 5]. That is, source statements are produced during the design and coding subcycle of Rayleigh curve. Thus, the integral of the manpower rate for this subcycle multiplied by the productivity gives source statements. From the above reasons, Putnam [Ref. 1] develops a

---

[6]total end product code / total effort to produce code

[7]Productivity rate * manpower = PR * Y

mathematical relationship, the SLIM software equation, which is an powerful tool for planning and evaluating the development effort life cycle. The software equation is:

$$S_s = \int_0^t \dot{S}_s \cdot dt = \int_0^t PR \cdot \dot{y}_1 \cdot dt = PR \cdot \int_0^t \frac{K}{t_d^2} \cdot t \cdot \exp\left(\frac{-1t^2}{t_d^2}\right) \cdot dt$$

$$S_s = C_k \, K^{1/3} \, t_d^{4/3} .$$

S_s = number of delivered source instructions

C_k = state of technology constant

K, t_d is equivalent to Rayleigh equation parameters

Putnam [Ref. 15] observed that the time at which the Rayleigh curve reaches its maximum value, t_d, corresponds to the time of system testing and product release for many software products. That is, normally peak manpower, $\dot{Y}$ max, exists as a function of development time(t_d),

$$\dot{Y}_{max} = K / \sqrt{e} \; t_d .$$

The area under the Rayleigh curve interval represents the total effort expended in that interval. Approximately forty percent of the area under the Rayleigh curve is to the left of t_d and sixty percent is to the right. By rearranging the software equation and applying a factor of .4 to reflect that the development effort, E, is approximately the first forty percent of the life cycle curve,

$$E \; (MM, MY) = .4K = .4 \left(\frac{S_s}{C_k}\right)^3 / \; t_d^4$$

By applying the burdened labor rate (average dollar cost per man year or man month of effort), the equation changes to cost

$$\text{Development cost} = (\$ / MY) \cdot 4 \left(\frac{S_s}{C_k}\right)^3 / t_d^4$$

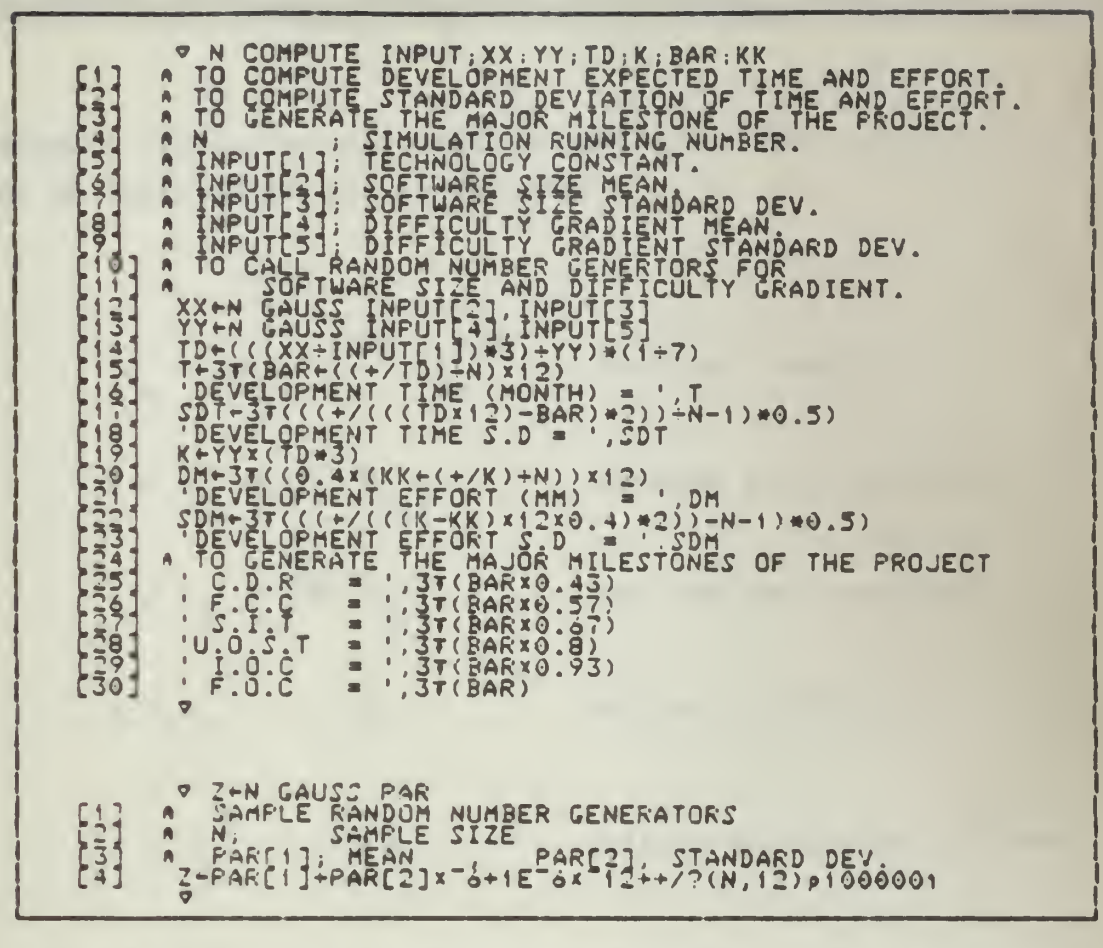
B. UNCERTAINTY ANALYSIS

Almost every factor entering into an estimate of effort and schedule is subject to some degree of uncertainty. The manager needs to portray the effects of the uncertainty associated with each of these factors. To get a reasonable estimate of uncertainty in life-cycle effort and development time, Monte Carlo simulation is used. Given C_K, S_S, σS_S, $|VD|$, and $\sigma|VD|$, Putnam [Ref. 2] obtained statistics on variability of the effort and develop time from several thousand samples. From the SLIM software equation and the difficulty gradient equation, we make the equation as follows:

$$t_d = \left(\frac{\left(\frac{S_s}{C_k}\right)^3}{|VD|}\right)^{1/7}$$

$$K = |VD| t_d^3$$

Also the standard deviation (σt_d, σK) obtained from a Monte Carlo simulation will be used to make a risk analysis profile. An APL program for this simulation analysis of estimating development time, effort, and the major milestone time is shown in Figure 4.2.



```
      ▽ N COMPUTE INPUT;XX;YY;TD;K;BAR;KK
[1]   ⍝ TO COMPUTE DEVELOPMENT EXPECTED TIME AND EFFORT.
[2]   ⍝ TO COMPUTE STANDARD DEVIATION OF TIME AND EFFORT.
[3]   ⍝ TO GENERATE THE MAJOR MILESTONE OF THE PROJECT.
[4]   ⍝ N          ; SIMULATION RUNNING NUMBER.
[5]   ⍝ INPUT[1]; TECHNOLOGY CONSTANT.
[6]   ⍝ INPUT[2]; SOFTWARE SIZE MEAN.
[7]   ⍝ INPUT[3]; SOFTWARE SIZE STANDARD DEV.
[8]   ⍝ INPUT[4]; DIFFICULTY GRADIENT MEAN.
[9]   ⍝ INPUT[5]; DIFFICULTY GRADIENT STANDARD DEV.
[10]  ⍝ TO CALL RANDOM NUMBER GENERTORS FOR
[11]  ⍝       SOFTWARE SIZE AND DIFFICULTY GRADIENT.
[12]   XX←N GAUSS INPUT[2],INPUT[3]
[13]   YY←N GAUSS INPUT[4],INPUT[5]
[14]   TD←(((XX÷INPUT[1])*3)÷YY)*(1÷7)
[15]   T←3↑(BAR←((+/TD)÷N)×12)
[16]   'DEVELOPMENT TIME (MONTH) = ',T
[17]   SDT←3↑(((+/(((TD×12)-BAR)*2))÷N-1)*0.5)
[18]   'DEVELOPMENT TIME S.D = ',SDT
[19]   K←YY×(TD*3)
[20]   DM←3↑((0.4×(KK←(+/K)+N))×12)
[21]   'DEVELOPMENT EFFORT (MM)   = ',DM
[22]   SDM←3↑(((+/(((K-KK)×12×0.4)*2))÷N-1)*0.5)
[23]   'DEVELOPMENT EFFORT S.D    = ',SDM
[24]  ⍝ TO GENERATE THE MAJOR MILESTONES OF THE PROJECT
[25]   ' C.D.R     = ',3↑(BAR×0.43)
[26]   ' F.C.C     = ',3↑(BAR×0.57)
[27]   ' S.I.T     = ',3↑(BAR×0.67)
[28]   'U.O.S.T    = ',3↑(BAR×0.8)
[29]   ' I.O.C     = ',3↑(BAR×0.93)
[30]   ' F.O.C     = ',3↑(BAR)
      ▽



      ▽ Z←N GAUSS PAR
[1]   ⍝  SAMPLE RANDOM NUMBER GENERATORS
[2]   ⍝  N;          SAMPLE SIZE
[3]   ⍝  PAR[1]; MEAN      ;      PAR[2]; STANDARD DEV.
[4]    Z←PAR[1]+PAR[2]×-6+1E¯6×12+.+?(N,12)ρ1000001
      ▽
```

Figure 4.2 APL Program for Development Time and Effort.

Linear programming is used to obtain the two most important answers in software development projects-minimum time and minimum cost. These "best possible" solutions also bound the range of reasonable solutions - all feasible solutions lie in between these extremes and are explicitly identified. The linear programming (LP) approach has another great advantage. It produces constrained optimal solutions. The SLIM model has introduced the most important software management constraints -- maximum cost, required delivery time, and staffing capability -- into the problem. This gives the program manager real control in his own domain of

resource allocation and control [Ref. 2]. The equations for the LP formulation are as follows:

$$S_s = C_k \, K^{1/3} \, t_d^{4/3}$$ software equation

$$K / t_d \leq \sqrt{e} \, \dot{Y}_{max}$$ maximum peak manpower

$$K / t_d \geq \sqrt{e} \, \dot{Y}_{min}$$ minimum peak manpower

$$K / t_d^2 < | D |$$ maximum difficulty

$$K / t_d^3 < | \nabla D |$$ maximum difficulty gradient

$$t_d \leq$$ contract delivery time

$$(\$ / MY) \; (.4 \; K) \leq$$ total budgeted amount for development

These can be solved with graphic or simplex methods. Figure 4.3 shows LP graphical feasibility solution using LOG-LOG transformation plot. [Ref. 2]

C. AVAILABLE OPTIONS

The SLIM function routine is composed of three options: top-level, what-if option, and implementation option. Figure 4.4 shows the SLIM methodology. [Ref. 4]

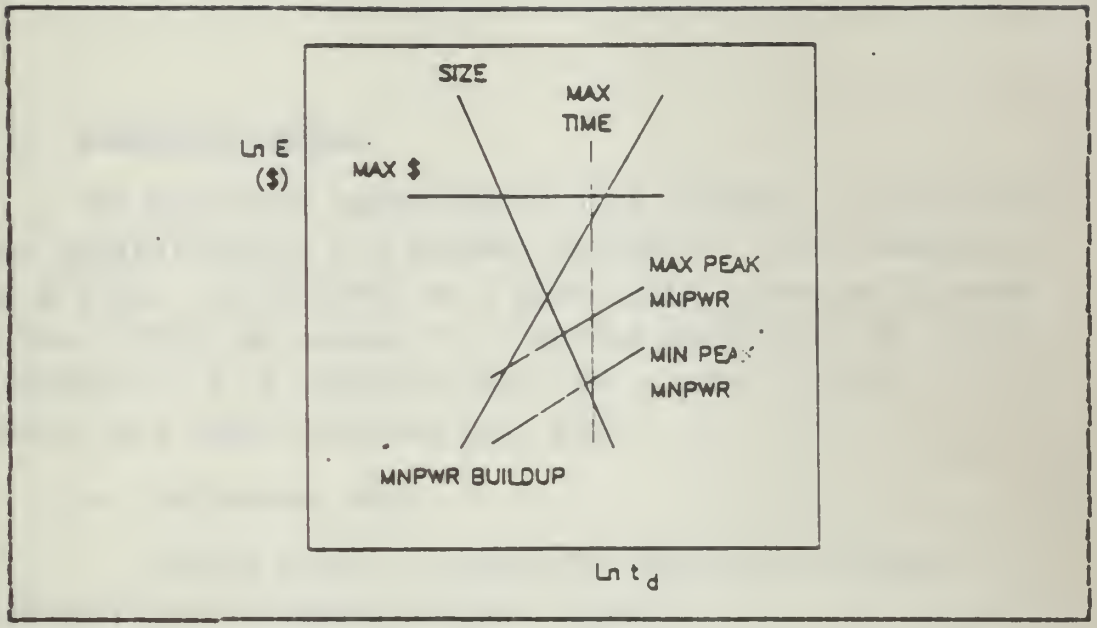


Figure 4.3 LP Graphical Solution [Ref. 2]

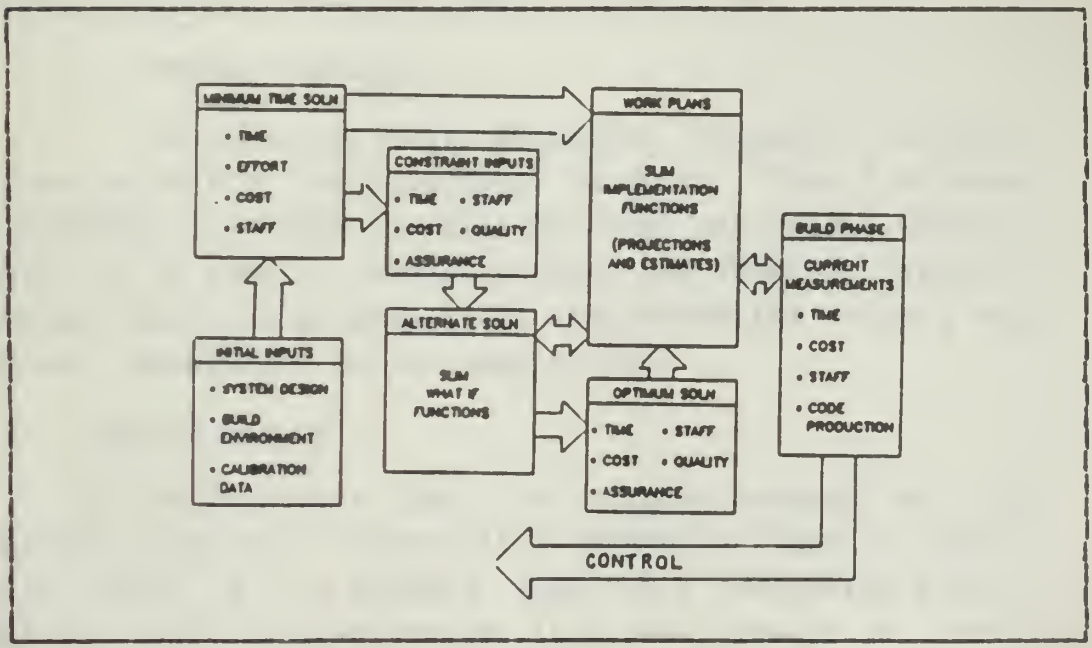


Figure 4.4 SLIM Diagram [Ref. 4]

1. Top-level Option

The top-level functions in SLIM include: Calibrate input, project input, and project estimate. The process by which a model is tailored to a particular class of systems with the intent of making it a better predictor is called calibration. It is important for the program manager to try to tailor his model from his past data.

a. Calibrate Input

Allows user to calibrate historic projects for productivity and manpower buildup indexes.

b. Project Input

Prompts user for all project estimate information and builds or modifies project data files.

c. Project Estimate

Provides all cost, schedule, manpower, quality and risk estimates for a software project. Once a minimum time solution has been established the management what-if options can be used to impose project constraints. When an acceptable solution is chosen the implementation options can generate a consistent set of work plans.

2. What-if Option

It is important for the program manager to try sensitivity analysis. Unless it is essential that the software be built in the minimum time, the estimator should consider alternative solutions that take longer but cost less. The what-if functions provide the ability to explore additional build strategies that take longer and could better suit the estimator's needs. The management what-if options include:

a. Design to Schedule

SIIS will automatically determine the minimum time schedule for which development of the system is feasible. This function may be used to set an alternative schedule for development. A new corresponding cost, effort and peak manpower will be provided.

b. Design to Cost

This function is used to allow the user to set a new cost for development. A new time schedule, level of effort and peak manpower will be generated.

c. Design to Effort

This function is used to allow the user to set a new level of effort for development. A new time schedule, cost and peak manpower will be generated.

d. Design to Risk

This function uses the trade-off law together with a user specified level of risk of not exceeding a required delivery date to generate an expected development time, level of effort, cost and peak manpower.

e. Design to Reliability

This function permits the user to input a specified Mean Time To Failure(MTTF). It then determines the appropriate development time, effort, cost and peak manpower to meet that MTTF together with the estimated number of errors and error rates.

f. Linear Program

This function uses the technique of linear programming to determine the minimum effort (and cost) or the minimum time in which a system can be built. The results

are based on the actual manpower, cost, and schedule constraints of the user, combined with the system constraints described earlier to yield a constrained optimal solution.

g. Best Bid

This function picks the best time-effort combination by maximizing the joint probability that the development time is greater or equal to a user- specified end date and the cost is less or equal to a user- specified cost.

h. Temporary Change

This function lets the user temporarily change up to nine parameters- title, start date, size, standard deviation on size, labor rate, uncertainty on labor rate, inflation rate, productivity index and manpower buildup index.

3. Implemention Option

Once the estimator has found a solution that best suits the estimator's requirement, the estimator will need a set of detailed plans to implement the solution. By periodically comparing progress of the development with the plan, the estimator has the means to control all phases of the software development from the beginning of the feasibility study through completion of operation and maintenance. The implementation options include:

a. Manloading

This function provides projections of the mean number of people (and standard deviation) that will be applied to the project throughout development. These projections are based on an optimal application of resources over time.

b. Cashflow

This function provides projections of the expected cashflow on a month-to-month basis throughout development of the system.

c. Life-Cycle

This function provides projections of the expected manpower and cashflow throughout the life-cycle of the system. Projections are provided on a monthly, quarterly, or yearly basis.

d. Risk Analysis

This function determines the probability of developing a system within a specified time or for a specified cost. it is very useful for strategic planning purposes by providing the associated with various time and cost decisions.

e. Work Breakdown

This function shows a breakout of effort devoted to specific skill categories as a function of the development schedule.

f. Effort Between Milestones

This function shows a breakout of effort devoted to principal activities as a function of the development schedule.

g. Milestones

Based on a predetermined total development time, this function provides a realistic schedule for the major milestones of the project.

h. Code

This function provides a rate of code production and cumulative code production for valid end product cost. Top down structured programming is assumed.

i. Gantt Chart

This function provides a Gantt Chart of activity phases and their respective milestones. It will reflect whether the design and coding is done in a top-down structured method or with a formal critical design review followed by a coding phase.

j. Front-End Estimates

This function provides low, average, and high estimates of the time and effort required for the feasibility study and design phase.

k. CPU Usage

This function provides a table of expected machine usage over the life cycle of the system, along with an estimate of total machine hours required for development.

l. Reliability

This function gives projections of error rate, cumulative errors created, found and fixed, and Mean Time To Failure month by month until a reliability of .999 is achieved.

m. Documentation

Based on the total system size and data from hundreds of similar software systems, a range of expected pages of documentation is given.

n. Summary of Development Costs

This function provides a summary of the major development costs for the system. These costs include labor costs and CPU costs over time and the total documentation cost.

o. Benefit Analysis

This function computes the benefit of the system required to amortize the cost of development and maintenance. It is based on the anticipated economic life of the system as well as the average rate of return for the organization.

The SLIM model furnishes an effective means to estimate and control the cost, schedule and manpower requirements of building and maintaining medium size to very large software systems. The program manager can tailor his estimates to meet all realistic constraints imposed on the development. After an intensive evaluation of the SLIM approach, the Department of Defense adopted the methodology as the standard macro estimating technique to be used on major software systems in all services and defense agencies.

V. PROCEDURE AND DATABASE

Most common methods of software costing start with estimation of the number of instructions. Boehm [Ref. 6] said the biggest difficulty in using today's algorithms in software cost models is the problem of providing sound sizing estimates.

Putnam [Ref. 1] suggests that in the systems definition phase, we do estimate a rough idea of the system size (source statements) and establish bounds on manpower effort and development years. In the requirement and specification phase, the number of files, reports, and application programs are good estimators of the size of the system and hence the time and effort required.

This paper is an investigation of ways in which we might be able to estimate the size of a new project, based on Putnam's existing data base of close to one thousand systems. Breakouts have been made of that data base by application type and the average size and the standard deviation of each of those application types has been calculated. The estimator can be approached in a Bayesian sense, such that if it is a scientific system, then it would fall somewhere in the range for a scientific system. This would be the Bayesian prior. Then this category is presented graphically, and the person asked whether it tends to be toward the low end of this category, toward the high end of this category, or low-middle or high-middle. By doing some statistical analysis, it is possible to come up with a weighted expected value and a new estimate of a standard deviation just by using the person's notional choice and the PERT-sizing formula. This would be a Bayesian likelihood estimate; it could be combined together with the prior using Bayes' theorem. The usefulness of the Bayesian approach to

statistical inference rests upon the usefulness of incorporating personal, subjective beliefs directly into the analysis.

From Bayesian inference and decision, it may be seen that the posterior probabilities are derived directly from prior probabilities and the likelihoods. It should again be recalled that the mean and standard deviation of a normally distributed random variable completely specify its probability function. Further, it is easily provided by means of the calculus that if the prior probability and the likelihood are both normally distributed, the posterior probabilities must be normally distributed. The reciprocal of the posterior variance (σ^2) is equal to the sum of the reciprocal of the prior variance (σ^2) and the reciprocal of the likelihood variance (σ^2), reflecting the fact that the two sources of information are pooled together. The posterior mean (E_2) is a weighted average of the prior mean (E_0) and the sample mean (E_1), the weights being the reciprocals of the respective variance [Pers. 16,17]. The equations are as follows:

$$\frac{1}{\sigma_2^2} = \frac{1}{\sigma_0^2} + \frac{1}{\sigma_1^2}$$

$$E_2 = \frac{\sigma_2^2}{\sigma_0^2} E_0 + \frac{\sigma_2^2}{\sigma_1^2} E_1$$

These posterior means and standard deviations will be used as SLIM input parameters.

A. QSM DATA BASE

QSM (Quantitative Software Management), Inc. is a company formed to help estimators solve critical cost and schedule control problems in business and government. The QSM data base is composed of one thousand systems from DOD, RADC (Rome Air Development Center), US Army Computer Systems Command, and various companies. Appendix G shows a sample for data format. The QSM data base is composed of ten types as shown in Table 3.

TABLE 3

QSM DATA BASE BY APPLICATION TYPE

Application Type	No. System	Avg system size
1 Micro code/ Firm ware data base	11	17435
2 Real-time data base	125	56870
3 Avionic data base	26	80186
4 system Software data base	75	75035
5 Command & Control data base	61	61452
6 Telecom data base	32	42179
7 Scientific data base	82	91265
8 Process Control data base	11	52780
9 Business data base	547	71693
10 Unknown application	1	140000
Mixed Application data base	971	69549

Table 4 shows that QSM data base divided by size range.

B. PROCEDURE

This idea is that by partitioning the data base according to statistical sub-groups, we estimate new project size using Bayesian inference. It may be possible to come close enough to the true value to give meaningful software estimates in early feasibility study phases of a major project. The procedure consists of five steps.

TABLE 4

QSM DATA BASE BY SIZE RANGE

Category	range (K)	possible spread range	# in DB	Ave. Size
1 Small	5K -15K	1K - 20 K	274	8296
2 Medium	15K- 35K	10K -40 K	241	24437
3 Medium Large	35K -75K	25k-125 K	207	53474
4 Large	75K- 200K	50K-250 K	179	120573
5 Very Large	200K-600K	200K-600K	62	318366
6 Extra Large	600K-1000K	500K-1000K	6	763060
7 Super Large	More than 1000K	750k-1500k	2	1200000

1. Determine Application Type

The different system types have unique properties which have to be considered so that parameters can be tuned to these properties. It is important to know what kind of software application type will be made. Using each application type's mean and standard deviation, we get statistical results from running the SLIM model. Appendix A shows the best case. In the best case, it is assumed that time is estimated within 2 percent and effort within 9 percent. Appendix F shows the extreme case. In the extreme case, time is estimated within 30 percent and effort within 80 percent.

2. Select Size Category and its Quantile

Project size is a major factor that determines the level of management control and the types of tools and techniques required on a software project. Initially the program manager determines the size category from Table 4 and, based on the possible spread range, he considers the overlap on both sides of the the category range. From application type and size range, the program manager can choose the proper mean and standard deviation of software size from Appendix

C. It will be used for Bayesian prior estimates. For example, let us select the category size of large. For this selection range of 75K-200K, the overlap range is graphically represented in Figure 5.1.

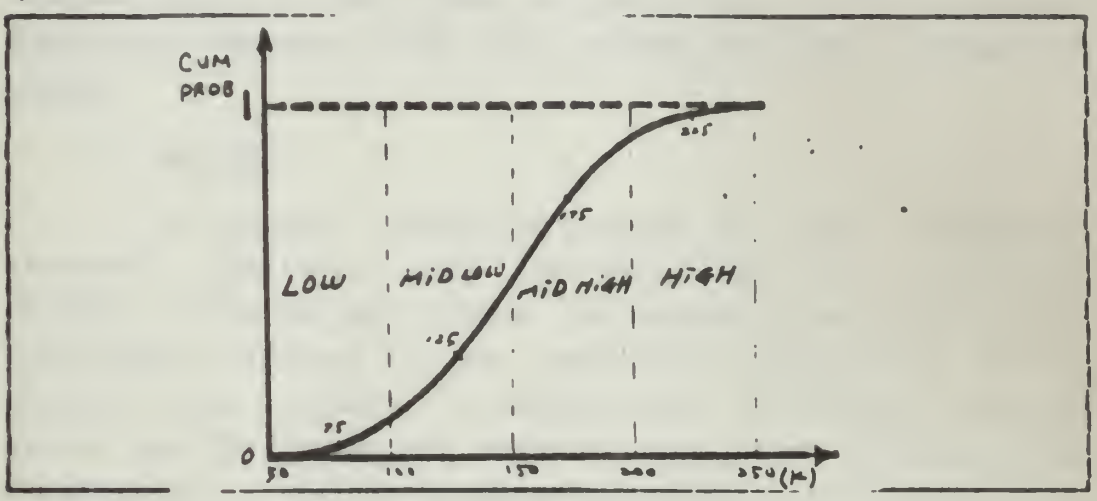


Figure 5.1 Where it Falls in the Range (Large).

Now select a size range from Low, Middle Low, Middle High, High. Assuming that the size is approximately normally distributed, we can use these as Bayesian likelihood estimates. These size ranges are also shown in Figure 5.1. The program manager believes it falls in the second quantile which we call "Mid Low". Using weighted means and the PERT-sizing formula, the project manager can get the mean and standard deviation of software size.

$$S = (50 + 4(125) + 250) / 6 = 133.3 \text{ (K)}$$

$$\sigma S = (250 - 50) / 6 = 33. \text{ (K)}$$

The other size categories and their respective quantiles are represented in Appendix D.

3. Compute New Mean and Standard Deviation

To get posterior mean and standard deviation as input parameters for SLIM, we use the Bayesian approach. We combine prior estimate from application's type and size and likelihood estimate from the program manager's subjective belief.

4. Run SLIM

The project estimates provide all cost, schedule, manpower, quality, and risk estimates for a software project. This output is used to determine the minimum time development strategy and its associated uncertainty. Once a minimum time solution is established the program manager should use the management what-if options to impose project constraints. When an acceptable solution is obtained, use the implementation reports and graphs to generate a set of work plans.

5. Analysis

To analyze the sensitivity of our solution to variations in the size, we use the temporary change routine. The report for temporary changes will contain a summary of the changes and a new minimum time solution based upon the changes. Results of the minimum time solution are output in three tables. The first table, the minimum time solution, gives a consistent set of management metrics (mean and standard deviation) for building the system in the shortest possible time. The second table, a sensitivity profile, shows how the management metrics change as a result of one and three standard deviation changes in the system size (mean). The third table, a consistency check, compares the estimator's solution with historical data for systems of comparable size in the QSM data base.

All the variables of the software equation are subject to some degree of uncertainty and the program manager must have a means of taking this into account in an effort development risk profile. Putnam [Ref. 1] suggests risk analysis is probably the most important aspect of any software system analysis. In an uncertain process, risk analysis measures the uncertainty to let us develop proper strategies to minimize the risk.

C. EXAMPLE

Let us examine a scientific type example in which a new stand-alone development project will be made. Initially the program manager believes that the project size category will be large. By observing the graphical representation of the range, the program manager subjectively estimates that the project falls into "Mid Low" range. For a scientific "Large" project, we can derive prior estimates for the mean and standard deviation of 117389 and 34508 respectively from Appendix C. Also we calculate likelihood estimates of mean and standard deviation of 133300 and 33000 respectively from Appendix D, where it falls in the "Mid Low" range. From Bayesian equations, we can get posterior estimates, as follows.

$$\frac{1}{\sigma_2^2} = \frac{1}{34508^2} + \frac{1}{33000^2} = \frac{1}{23850^2}$$

$$\bar{x}_2 = \frac{23850^2}{34508^2}\ 117389 + \frac{23850^2}{33000^2}\ 133300 = 125700$$

These estimates will be used as SLIM input parameters. We assume that this project uses average manpower and productivity index. From running the SLIM model, we get the minimum time solution. Knowing the minimum development time and standard deviation give the program manager a framework from which to plan and control the software development process. From Table 5, for the minimum time solution, the program manager should consider the estimate having a 50 percent chance of being realized upon completion of the system. When this expected value is added to its standard deviation, it gives a new value that has an 84 percent chance of being realized. Also the percent of expected value of the standard deviation is a major factor. For this example it is as follows:

$$\sigma t_d \;/\; t_d = 2.1 \;/\; 23.4 = .089$$

$$\sigma E \;/\; E = 139 \;/\; 516.5 = .269$$

This shows that development time is estimated within 9 percent and effort within 27 percent in one standard deviation. The other range results are shown in Appendix E. Appendix F shows it for business applications.

Table 6 shows the corresponding expected values for time, effort and cost by increasing one and three standard deviation of system size. Also it shows that a 99 percent confidence interval of development time is between 16.2 month and 28.2 month.

Table 7 shows that if the values shown are within plus or minus 45 percent of the mean for systems of comparable size in the data base, the table labels them as "in normal range". Otherwise the values will be reported as either greater than or less than the normal range.

TABLE 5

MINIMUM TIME SOLUTION

TITLE: Thesis Example

DATE: 20-SEP-1985
TIME: 11:27:46

MANAGEMENT METRIC	EXPECTED VALUE (50% PROBABILITY)	STD DEV
SYSTEM SIZE (STATEMENTS)	125700	23850
MINIMUM DEVELOPMENT TIME (MONTHS)	23.4	2.1
DEVELOPMENT EFFORT (MANMONTHS)	516.5	139.0
DEVELOPMENT COST* (x 1000 $)		
(UNINFLATED)	2178	668
(INFLATED)	2396	747
PEAK MANPOWER (PEOPLE)	34	7

TABLE 6

SENSITIVITY PROFILE

	SOURCE STMTS	MONTHS	MANMONTHS	UNINFLATED COST (X 1000)
-3 STD DEV	54150.	16.2	162.	675.
-1 STD DEV	101850.	21.3	386.	1607.
EXPECTED	125700.	23.4	516.	2178.
+1 STD DEV	149550.	25.1	633.	2635.
+3 STD DEV	197250.	28.2	903.	3762.

Figure 5.2 shows the major milestone time for the current solution as a statistical expected value, i.e. , the value that statistically has a 50 percent probability of not being exceeded upon completion of the main program.

Table 8 shows the probability ranges for effort, cost, and time from 1 percent to 99 percent --a broad enough band to cover all realistic possibilities.

TABLE 7

CONSISTENCY TABLE

MANAGEMENT METRIC	VALUE	ASSESSMENT
TIME (MONTHS)	23.4	IN NORMAL RANGE
EFFORT (MANMONTHS)	516	IN NORMAL RANGE
AVERAGE STAFFING (PEOPLE)	22	IN NORMAL RANGE
PRODUCTIVITY (LINES/MM)	243	IN NORMAL RANGE

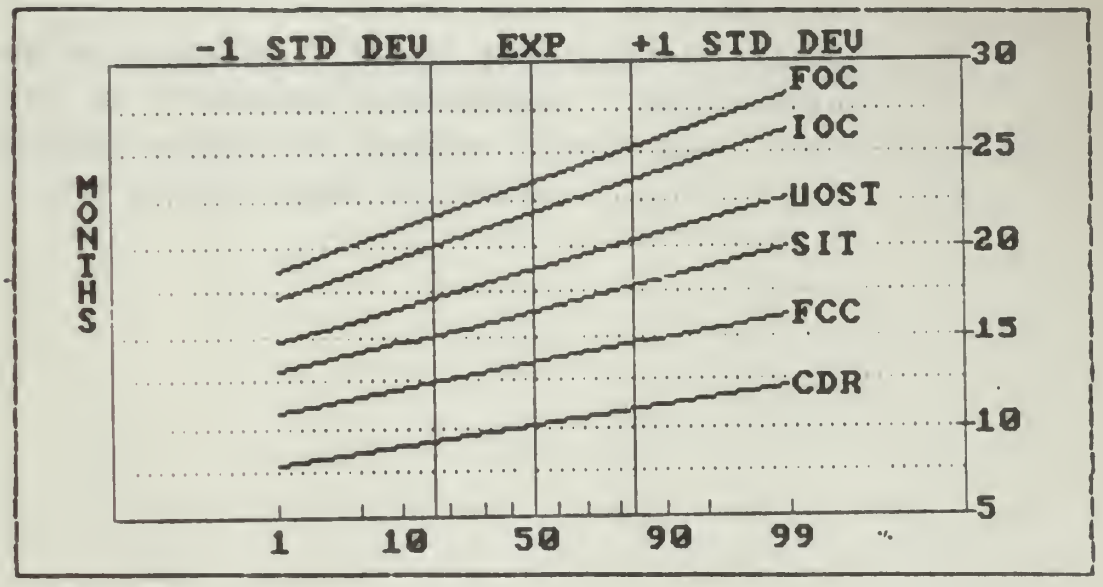


Figure 5.2 Risk Profile (time).

From the tables and graph, the program manager determines the probability that other times, effort, and costs will not be exceeded. Development time and milestones are the most sensitive elements in the process. Milestones scale linearly, meaning that if the project manager is late on an

TABLE 8

PROBABILITY PROFILE

PROBABILITY	MANMONTHS	UNINFLATED COST (X $ 1000)	TIME (MONTHS)
1 %	193	625	18.6
5 %	288	1080	20.0
10 %	338	1322	20.8
20 %	400	1614	21.7
30 %	444	1828	22.3
40 %	481	2009	22.9
50 %	516	2178	23.4
60 %	552	2347	23.9
70 %	589	2528	24.5
80 %	633	2740	25.2
90 %	695	3033	26.1
95 %	745	3276	26.8
99 %	840	3751	28.2

early milestone, the project manager will very likely be late on succeeding milestones. It is important for the program manager to remember Brooks' law: Adding manpower to a late project makes it later.

VI. CONCLUSION AND RECOMMENDATIONS

In order for the program manager and his support team to evaluate a software cost proposal, they must have a detailed understanding of the process that developed the cost estimate and should have some traceability to the variances or sensitivity of the estimating components. It is important for the program manager to recognize the similarities/differences between the defense system life cycle and the other commercial development life cycle. The program manager should compare various cost models for alternative estimates.

The life-cycle curve can be mathematically represented by the Rayleigh equation, which is a good management tool. For planning and evaluating the development effort life cycle, the SLIM model is a very powerful tool.

It is hard to estimate software size in the early phase of system development. But the program manager can estimate the new project size using Bayesian inference. To get a Bayesian estimate of software size, the program manager combines a prior estimate from application's type and category size with a likelihood estimate from the program manager's subjective belief. When combined with the capabilities of the SLIM model, the development time is estimated within 15 percent and effort within 30 percent in one standard deviation.

This method could be a fruitful way to get at the early estimating problem in a gross way, yet good enough to get in the "ball park" of what people need at that phase of the project.

Future research should focus on the relationship between cost attribute factors and system application types.

APPENDIX A

DISTRIBUTION BY APPLICATION TYPE(BEST CASE)

Application Type	Development Time		Effort	
	Mean.	Std. Dev	Mean.	Std.Dev
1 Micro code/Firm ware	10.00	.21	16.0	1.3
2 Real-time	16.59	.38	174.0	15.5
3 Avionic	19.20	.44	282.9	25.4
4 system Software	18.69	.41	257.0	22.3
5 Command & Control	17.20	.38	192.7	16.7
6 Telecom	14.61	.30	108.1	8.8
7 Scientific	20.31	.45	334.3	28.8
8 Process Control	16.06	.37	156.1	14.3
9 Business	18.29	.40	243.8	20.5
Mixed Application	18.05	.38	233.5	19.2

APPENDIX B

DISTRIBUTION BY APPLICATION TYPE (EXTREME CASE)

Application Type	Development Time		Effort	
	Mean.	Std. Dev	Mean.	Std.Dev
1 Micro code/Firm ware	10.43	3.64	24.8	20.2
2 Real-time	21.37	7.10	498.4	420.8
3 Avionic	22.04	7.15	553.7	457.6
4 system Software	21.64	6.85	515.1	411.9
5 Command & Control	19.65	6.70	400.6	351.0
6 Telecom	16.33	5.49	202.5	173.5
7 Scientific	25.80	10.12	975.8	912.6
8 Process Control	18.17	5.25	276.9	206.4
9 Business	21.73	7.15	529.9	427.1
Mixed Application	20.98	7.36	477.6	417.7

APPENDIX C

DISTRIBUTION BY APPLICATION TYPE AND SIZE BIN

	1 - 15 K			15K-35K			35K-75K		
App type	No.	Mean	St Dev.	No.	Mean	St Dev.	No.	Mean	St Dev
1 Micro / firm ware	7	55417	3985	3	23333	4714	0	0	0
2 Real-time	45	7943	4388	31	22599	4854	27	51507	13801
3 Avionic	5	8400	3426	7	23686	5112	6	52507	13801
4 System software	20	7020	3761	19	25466	5421	16	53513	8877
5 Command & control	23	9676	4724	16	26587	4997	8	47518	7987
6 Telecom	15	8931	3611	5	25700	6112	8	50625	11131
7 Scientific	22	8352	4108	18	23171	5606	22	50156	7194
8 Process control	4	9322	2502	2	24247	7746	3	51333	6342
9 Business	133	8409	3828	140	24641	5715	117	55528	11839
Mixed	274	8298	4004	241	24437	5626	207	53474	10967

DISTDIBUTION BY APPLICATION TYPE AND SIZE BIN

	75K-200K			200K-600K			600K-1000K		
App type	No.	Mean	St Dev.	No.	Mean	St Dev.	No.	Mean	St dev
1 Micro / firm ware	1	83866							
2 Real-time	16	120101	40130	5	413115	152087	1	699000	
3 Avionic	5	102600	16704	3	351667	28193			
4 System software	13	111053	26872	5	264820	69050	2	689681	22679
5 Command & control	10	113153	38547	4	397250	124289			
6 Telecom	3	143318	41904	1	252320				
7 Scientific	13	117389	34508	4	313358	102764	2	900000	99999
8 Process control	2	170400	5400						
9 Business	115	122192	33535	40	304981	87052	1	700000	
10 Unknown	1	140000							
Mixed	179	120573	34540	62	318366	102241	6	763060	113563

APPENDIX D

DISTRIBUTION BY RANGE AND QUANTILE

Description	Possible overlap range	Possible Quantile "Low"	" Mid. Low"	" Mid High"	"High"	Std Dev.
1 Small	1K -20K	5.5	8.5	11.8	15.2	3.2
2 Medium	10K- 40K	17.5	22.6	27.8	32.3	5
3 Medium large	25K -125K	50	66.7	87.5	100	16.7
4 Large	50K- 250K	100	133.3	166.7	200	33.
5 Very Large	200K-600K	300	366.7	433.3	500	66.7
6 Extra Large	500K-1000K	625	708.3	791.7	875	83.3
7 Super Large	750K-1500K	812.5	854.2	895.8	937.5	125

APPENDIX E

SCIENTIFIC DATA BASE EXPECTED SIZE, TIME, AND EFFORT

Size(mean, %), Time(mean, dev), and Effort(mean, dev)

(1-15k)	" Low"	"Mid Low"	"Mid High"	"High"
size %	6563 .38	8445 .30	10533 .24	12628 .20
time dev.	6.5 1.3	7.3 1.1	8.1 .9	8.7 .8
effort dev.	4.9 2.4	6.6 2.6	8.7 2.8	10.8 3.0
(15-35k)	" Low "	" Mid Low"	"Mid High "	" High "
size %	20013 .1[illegible]	23836 .16	25766 .14	28272 .13
time dev.	10.7	[illegible].3 .9	11.9 .8	12.4 .8
effort dev.	22.7	7.5	42.0 9.0	51.4 10.3
(35-75k)	"Low"	Low"	"Mid High"	"High"
size %	501[illegible]	[illegible].9 .12	56021 .11	57984 .11
time dev.	15.8	[illegible].1 1.0	16.6 1.0	16.8 .9
effort dev.	145.5 [illegible]	157.2 30.2	171.9 31.5	180.8 32.3
(75-200K)	"Low "	Mid Low"	"Mid High "	"High"
size %	108306 .2[illegible]	125700 .19	143128 .17	160539 .15
time dev.	22.2 2.3	23.4 2.1	24.8 1.9	26. 1.8
effort dev.	428.9 131.3	516.5 139	607.7 146.6	702.1 154.1
(200-600K)	"Low"	"Mid low"	"Mid High"	"High"
size %	303957 .18	350877 .15	397796 .14	447166 .13
time dev.	34.2 2.9	36.4 2.7	38.4 2.6	40.3 2.4
effort dev.	1805.6 421	1922.3 447	2251.5 472	2610 500

APPENDIX F

BUSINESS DATA BASE EXPECTED SIZE, TIME, AND EFFORT

Size(mean, %), Time(mean, dev), and Effort(mean,dev)

(1-15k)	" Low"	"Mid Low"	"Mid High"	"High"
size %	6678 .37	8463 .23	10444 .23	12431 .20
time dev.	6.6 1.2	7.3 1.0	8.1 .9	8.7 .8
effort dev.	5.0 2.4	6.6 2.6	8.6 2.8	10.6 2.9
(15-35k)	" Low "	"Mid Low"	"Mid High "	" High "
size %	20594 .18	23468 .16	26447 .14	28996 .13
time dev.	10.8 .9	11.4 .9	12.0 .8	12.5 .8
effort dev.	24.4 6.4	33.8 7.9	44.5 9.4	54.3 10.7
(35-75k)	"Low"	"Mid Low"	"Mid High"	"High"
size %	53674 .18	59264 .16	66250 .15	70442 .14
time dev.	16.3 1.4	17.0 1.3	17.8 1.2	18.3 1.2
effort dev.	163 41.9	188.2 44.5	220 47.5	239.2 49.2
(75-200K)	"Low "	" Mid Low"	"Mid High "	"High"
size %	110918 .21	127852 .18	144787 .16	161722 .15
time dev.	22.2 2.4	23.6 2.0	24.9 1.9	26.1 1.8
effort dev.	441.5 130.8	527.3 138	616.3 146	708.4 152.8
(200-600K)	"Low"	"Mid Low"	"Mid High"	"High"
size %	301841 .18	343863 .15	385884 .14	427905 .12
time dev.	34.1 2.6	3o.4 2.6	37.9 2.5	39.6 2.4
effort dev.	1589 400	1871 423	2164 446	2466 469

APPENDIX G

QSM SAMPLE DATA

Run Date: 06-25-1985 PROJECT DETAIL REPORT Page: 1
Run Time: 11:00:59 Project: GROSS MARGIN ANLYS

Date: 05/21/85 Project name : GROSS MARGIN ANLYS
Organization: System number: 964

SIZING INFORMATION
Total Source Statements: 23703

Name of Language	Type	% of Total Size
COBOL	High Level	22 %
NATURAL	Non-Procedural	71 %
JCL	High Level	7 %
_____	_____	___ %
_____	_____	___ %

% of Total Source Statements
Brand New (ss): 100 %
Modified (ss): ___ %
Unmodified (ss): ___ %

TIME-EFFORT-STAFFING

Feasibility Study:	mos	.5 mm		
Functional Design:	4 mos	3.5 mm	1.5 Pk Mpwr	_____ Overlap (mos)
Main Software Build:	3 mos	10 mm	7 Pk Mpwr	_____ Cost
Operations + Maint.:	3 mos	2.5 mm		

FOC Date: 01/85

OVERRUN/SLIPPAGE (MAIN SOFTWARE BUILD)
Time Slippage: _____ mos Cost Overrun: _____ (mm)

QUALITY (MAIN SOFTWARE BUILD)
Errors (SIT-FOC): _____ MTTF (1st mo oper): _____ days
Errors (1st mo after FOC): 13

PROJECT CONSTRAINTS (MAIN SOFTWARE BUILD)
Cost: _____ People: 7
Time: 8 mos Computer: Severe

ENVIRONMENT
Average Personnel Skill Experience

Overall:	Medium	Similar Systems:	Medium	Languages:	Medium
Computer:	Medium	Methodologies:	_____	Software Aids:	High
Mgmt Team:	High	Staff Turnover:	_____	Tools + Utils:	Good
Response Time:		5 Minutes - 1 Hour			

Development Computer: IBM 3033
Memory Occupancy: ___ %
Real Time Code: 0 %
Requirements Change: 2 %

APPLICATION
Application Type: Business
System Features:
Data Base

SYSTEM DESIGN COMPLEXITY
Algorithms mostly known but logic designed from scratch. Interfaces were straight forward.

CALCULATED FIELDS
Productivity Index: 21 Manpower Buildup Index: 6
Total Developed, Delivered, Executable Source Lines of Code: 23703

USER FIELDS
1)
2)
3) USA
4) HI CONFID

PROJECT DESCRIPTION
ONLINE REPORT REQUESTING WITH 3 SCREENS OF REPORT SELECTION CRITERIA AVAILABLE

REPORTS SHOW GROSS MARGIN INFORMATION FOR SEGMENTS OF BUSINESS NIGHT BATCH RUN

FACTORS THAT HAD A POSITIVE OR NEGATIVE IMPACT ON THIS PROJECT
-LACK OF CLEAR INFORMATION ON DATA WHICH IS USED TO GENERATE REPORTS LED TO INCORRECT ASSUMPTIONS ERRORS WERE FOUND DURING TESTING WHICH CAUSED THIS PHASE

TO TAKE LONGER THAN PLANNED.
-ONE BATCH PROGRAM NEEDED ABNORMALLY LARGE TABLE REDESIGN NECESSARY TO MEET PERFORMANCE CRITERIA
-TIGHT SCHEDULE +USER PROJECT MONITOR DID A THROUGH TESTING JOB
+ TEAM WORKED AT NIGHT TO GET ACCEPTABLE TURNAROUND ON THE CPU THIS GOT THE JOB DONE BUT HAD A ADVERSE EFFECT ON THE TEAM PHYSICALLY

TOOLS, UTILITIES, COMPUTER BASED AIDS & METHODOLOGIES USED ON THIS PROJECT
TOOLS:
SAS TEST & DEBUG TOOL
NATURAL
ADABASE
MAYFLOWER SDM SYSTEM DEVELOPMENT METHODOLOGY

LIST OF REFERENCES

1. Putnam, Lawrence H. , Tutorial on Software Cost Estimating and Life-cycle Control : Getting the Software Numbers, IEEE, 1980.

2. Putnam, Lawrence H. , Reference notes for the SLIM Software Cost Estimating Course, QSM, 1985.

3. Masters, Thomas F. , "Overview of software cost estimating at the National Security Agency," Journal of International Society of Parametric Analysts, Vol. 5, No. 1, Jan 1985.

4. Putnam, Lawrence H. , SLIM Users Manual, QSM, 1984.

5. Boehm, Barry W. , Software Engineering Economics, Prentice-Hall, 1981.

6. Boehm, Barry W. , "Software Engineering Economics," IEEE Transactions on Software Engineering, Vol SE-10 No 1, Jan 1984.

7. Fairley, Richard E. , Software Engineering Concepts, McGraw-Hill, 1985.

8. Second Software Life Cycle Management Workshop, 78CH1390-4C, IEEE Computer Society, 1978.

9. Boehm, B. W. , "Software Life Cycle Factors," Handbook of Software Engineering, Van Nostrand Reinhold Company, pp. 494-518, 1984.

10. Stanley, Margaret, "Software Cost Estimating," Journal of International Society of Parametric Analysts, Vol. 4, No. 3, September 1984.

11. Rome Air Development Center, 1981. An Evaluation of Software Cost Estimating Models, by R. Thibodeau, RADC-TR-81-144, 1981.

12. Wolverton, R. W. , "Software Costing," Handbook of Software Engineering, Van Nostrand Reinhold Company, pp. 469-493, 1984.

13. Bruce Phillip and Pederson, Sam M. , The Software Development Project (Planning and management), John Wiley, 1982.

14. Putnam, Lawrence H. and Wolverton, Ray W. , Tutorial Quantitative Management: Software Cost Estimating, IEEE, 1977.

15. Pressman Rogers, Software Engineering: A Practitioner's Approach, McGraw Hill, 1982.

16. Morgan, Bruce W. , An Introduction to Bayesian Statistical Decision Processes, Prentice-Hall, Inc, 1968.

17. Box, George E. and Tiao, George C. , Bayesian Inference in Statistical Analysis , Addison-Wesley, 1973.

BIBLIOGRAPHY

Bryan William, Software Configuration Management IEEE, 1980.

Demarco Tom, Controlling Software Projects, Yourdon Press, 1982.

Lapin Lawrence, Quantitative Methods for Business Decisions Harcourt Brace Jovanovich, Inc. , 1976.

Stewart, Rodney D. , Cost Estimating, John Wiley & Sons, 1982.

Ware, Willis H. and Patrick, Robert L. , Perspective on oversight management of software development projects, RAND, July 1983.

Washburn, Alan R. , NOMPAS - a Bayesian Procedure for selecting the Greatest mean, Naval Postgraduate School, Jun 1982.

Winkler, Robert L. , Introduction to Bayesian Inference and Decisions, Holt Rinehart and Winston, Inc, 1972.

INITIAL DISTRIBUTION LIST

No.		Copies
1.	Defence Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93943-5100	1
4.	Professor D. C. Boger, Code 54 BK Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	3
5.	Professor M. G. Sovereign Code 55 ZO Department of Operations Research Naval Postgraduate School Monterey, California 93943-5100	2
6.	President. Lawrence H. Putnam Quantitative Software Management, Inc. 1057 Waverley way Mclean, VA 22101	2
7.	Professor Tung X. Bui, Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5100	1
8.	LT Park, In Kyoung 443-21 Young Hyun Dong, Nam-Gu, Inchon 160-02, Seoul Korea.	9
9.	Library Naval Academy Jin Hei 602, Seoul Korea.	2
10.	Naval Headquarters P.C. BOX 201-01 Sin Kil 7 Dong, Young dong Po Gu, Secul 150-09, Korea	3
11.	Chairman. Computer Center P.C. BOX 201-01 Naval Headquarters Sin Kil 7 Dong, Young Dong Po Gu, Seoul 150-09, Korea	1
12.	Chairman. Computer Center P.C. BOX 77 Gong Neung Dong, Do Bong Gu, Seoul 130-09, Korea	1
13.	Chairman. Computer Center Airforce Academy Dae Bang Dong, Young Dong Po Gu, Seoul 151, Korea	1